



Measurement and Communication of Software Risk

SQNLDR Patrick Redmond

SCI-DGTA

2010 ADF Software Symposium





Overview

- Background
- Sources of Risk
- Summary of the Problem
- How can we communicate software risk?
- How can we measure software risk?
- Where to from here?





Disclaimer

DGTA does not yet have a formal policy on the measurement and communication of risk arising from shortfalls in the assurance of aviation software integrity.

This presentation represents my views on this issue, but is not DGTA policy.





Background





Pop Quiz Hotshot

- You are the commander of a transport squadron that has been tasked to fly into hostile airspace to evacuate Australians.
- There are surface to air threats in the area.
- Your aircraft is fitted with a self protection system that is implemented in software.
- But, there is a problem:
 - A possible failure mode of the system could cause Catastrophic failure of the airframe.
 - Very little is known about the software.
- What do you do?
 - Note: you can't shoot the hostage.





Decision Making Time

- The following will constrain your decision:
 - Not going is not an option.
 - There is not enough time to fix the software.
- Your decision should be based on risk.
 - The risk of being shot down, vs
 - The risk of the software driven self protection system causing a crash.
- Both consequences are the same: Catastrophic.
- The distinguishing factor is likelihood.
- Perhaps you would tolerate the risk that has the lowest likelihood.
 - The likelihood of being shot down, vs
 - The likelihood of software failure.





Bring in the Engineers

- At this point, you (the operational commander) call the engineers and ask:
 - How likely is it that the software will fail and cause the aircraft to crash?

- As the operational commander:
 - How would you like the engineer to respond to this request?
 - How sure do you need the engineer to be?





Back to Reality

- This scenario is overly simplified and overly dramatic.
- But similar (more complex) issues often arise:
 - How do you make an informed decision to trade safety related software risk for cost, schedule or capability benefits?
- There's a very good chance that the engineer approached by the operational commander would be someone in this room.
 - How would you respond?





Real Life Examples

- Primary Flight Display Software (Catastrophic)
 - Most DO-178B Level A objectives satisfied.
 - A couple of key objectives not satisfied, or only partially satisfied.
 - Schedule pressure on release of the software.
 - Would you recommend release?
- Software with Catastrophic Consequences
 - Details not releasable to this forum.
 - Some key Level D objectives were not satisfied.
 - Traceability between system and software requirements.
 - Testing of software requirements.
 - Capability was needed for operations.
 - Would you recommend release?





Real Life Examples

- Software that controls a function that places the general public at risk.
 - OEM did not consider the software safety critical.
 - No access to data.
 - Capability required by the ADF.
- Flight Control System
 - No fault tolerance or redundancy.
 - Poorly controlled software development.
 - Capability required by the ADF.





Breaking the Problem Down

- What is the source of software risk?
 - What is it that is of concern?
- How do we meet stakeholder needs in communicating software risk?
 - Software doesn't fit neatly into traditional risk management paradigms.
 - How we need to communicate the risk drives how it needs to be measured.
- How do we measure software risk?
 - Risk analysis generally requires likelihood.





Sources of Risk





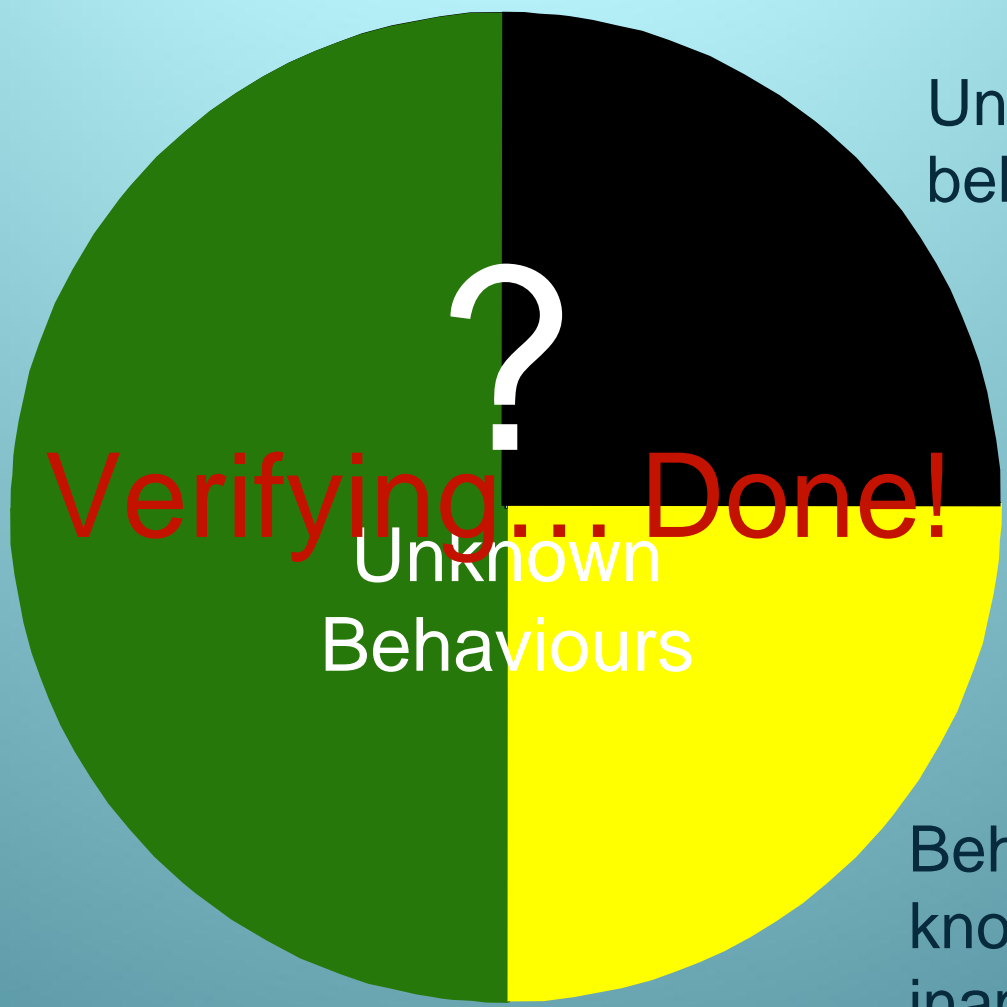
Software Risk in General

- In this presentation, I'm talking about safety risk.
 - Not cost, schedule or capability risks.
- How does software present a safety risk?
 - It performs safety related functions, provides safety related information, or controls safety related equipment, and
 - it contains a fault that, when exposed, will lead to non-benign behaviours.
- Sometimes we are aware of software faults, sometimes we are not.
 - Software faults are like landmines, you don't know they are there until you step on them.





Software Risk



Behaviours known to be appropriate.

Unknown behaviours.

Verifying... Done!

Unknown Behaviours

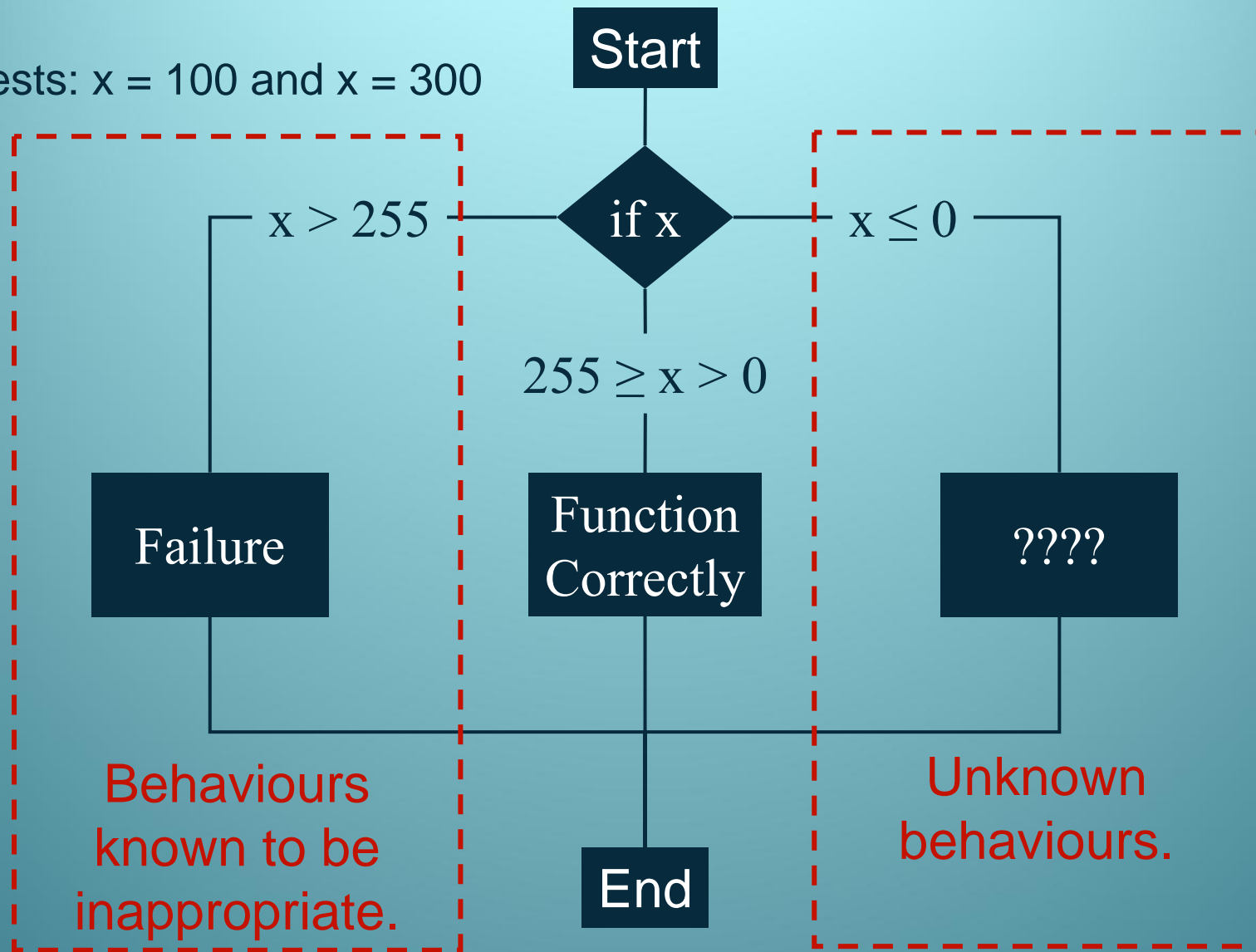
Behaviours known to be inappropriate.





Software Risk – Another View

Two Tests: $x = 100$ and $x = 300$





Sources of Software Risk

- As the robustness of verification activities increases, the proportion of unknown behaviours decreases.
- Behaviours that are known to be appropriate are not sources of risk.
 - i.e. the software does what it is supposed to do.
- Software risk arises from behaviours that are known to be inappropriate and unknown behaviours.





Software Assurance and Residual Risk

- Behaviours known to be inappropriate are straightforward to manage.
 - Done everyday through the problem reporting system.
- How do we manage unknown behaviours?
 - Every software item has a degree of unknown behaviours.
 - This includes civil aircraft, the space shuttle, nuclear reactors...
 - The issue is whether the degree of unknown behaviours is acceptable.
 - The higher the criticality, the less unknown behaviours are tolerated.
 - This is software assurance.
 - But note that there is a second order protection against unknown behaviours (fault tolerance) that is implemented through system safety and software safety programs.





Risk Treatment Options

- Decisions on risk treatment for behaviours that are known to be inappropriate is made on a case by case basis.
 - As the behaviours are known, the effects are easy to quantify.
- Decisions on risk treatment options for unknown behaviours are more complicated.
- If the four airworthiness requirements for software are satisfied, the TAA retains the residual risk presented by unknown behaviours.
 - i.e. the TAA is satisfied that complying with appropriate standards is sufficient to demonstrate that the degree of unknown behaviours is acceptable.
- But what happens if an airworthiness requirement is not satisfied?





System Safety

Software Safety

Software Assurance

Software Development





Specific Sources

- Common examples of shortfalls that result in the potential for unknown behaviours:
 - Insufficient Testing
 - Structural coverage not complete
 - Robustness criteria satisfied
 - Irrelevant Testing
 - Not linked to requirements
 - Non-Interference
 - Limited proof that you haven't broken what you haven't changed
 - Traceability
 - Can't account for all object code
 - Low Level Requirements Definition
 - Coders forced to make design assumptions





The Problem

- Unknown behaviours within safety related aviation software present a risk to the airworthiness of an aircraft.
- How can the technical community communicate the level of such risk?





Before we get too far into this...

- The measurement and communication of software risk is a last resort.
- DGTA expects that recognised practices will be applied.
- The discussion of measurement and communication of risk that you are about to see is intended only to describe methods for informing risk treatment decisions.
- It enables a risk authority to relatively trade safety for cost, schedule and capability.
- By the very nature of the problem, they cannot be anything other than low fidelity estimates (i.e. they have a high potential for error).
- As such, the methods about to be discussed cannot be used as an argument for the safety of software.
 - i.e. you should never argue the safety of your software on the basis that the “likelihood of failure” is sufficiently low.





How can we communicate software risk?





Risk Management

- AS/NZS 4360:2004 Risk Management
 - Risk is “the chance of something happening that will have an impact on objectives.”
 - “Note 2: Risk is measured in terms of a combination of the consequences of an event and their likelihood.”
- Defence Aviation Safety Manual
 - “Risk is the chance of something happening that will have an impact upon objectives. It is measured in terms of consequences and likelihood.”





What do operators expect?

- Operators use *Aviation Risk Management (AVRM)* to manage risk.
- AVRM is a tailoring of AS/NZS 4360 to the ADF Aviation domain.
- Technical Airworthiness risks are managed within the system safety domain.
 - But must be converted to AVRM prior to communication to operators.
 - This enables operators to trade technical airworthiness risk for other risks (capability, mission, public image, etc).
- At this point in time, there is a requirement to communicate to operators in AVRM terminology





Current Communication Requirements

- Risk must be referred to operators in AVRM terminology.
- This requires assignment of a risk level:
 - Consequence
 - Likelihood
- For safety risks resulting from unknown behaviours, the consequence is driven by the worst credible failure condition.
 - Behaviours are unknown: conservatively assumed that they may contribute to worst credible failure condition.
- A likelihood must also be assigned.





AVRM Likelihoods

ADF AVRM **Definitions & Risk Levels**

Likelihood	Likelihood Definition
LIKELY	Expected to occur during the activity under consideration.
PROBABLE	Could occur during the activity under consideration.
POSSIBLE	Occurrence conceivable but only expected infrequently during the aviation system life (nominally 20 years).
IMPROBABLE	Occurrence conceivable but only expected on a few occasions during the aviation system life (nominally 20 yrs).
RARE	Occurrence conceivable but expected no more than once during the aviation system life (nominally 20 years).





How do we assign a likelihood?

- It may be possible to assign a likelihood to behaviours known to be inappropriate.
 - Based on likelihood of set of inputs required to expose the fault. Difficult but possible.
- How do you assign a likelihood to the presence and exposure of unknown behaviours?
 - General consensus is that there is no sound engineering basis for doing this.
 - Obviously a likelihood can be assigned, but there is a question of correctness.
- We'll deal with the how later, for now it is enough to know that:
 - current risk management practices requires a likelihood
 - a likelihood cannot be correctly assigned when the source of risk is uncertainty in software behaviours





Options for Communicating Software Risk

- Option 1:
 - Assign a likelihood, but ensure the flaws are communicated (current practice).
- Option 2:
 - Dispense with likelihood, just assign a level.
- Option 3:
 - Replace likelihood with uncertainty.
- Option 4:
 - Take the FAA approach: acceptable or unacceptable.





Option 1: Use Likelihood

- Overview
 - Assign a likelihood based on a qualitative estimate.
 - Communicate uncertainty in estimate.
- Strengths
 - Conforms to AVRМ.
 - Risk level not necessarily correct, but understandable.
- Weaknesses
 - Potential to mislead.
 - Qualitative estimate may be grossly incorrect.
 - Potential for misuse of assigned likelihood.





Option 2: Just a Level

- Overview
 - Assign a risk level based solely on the severity of failure conditions.
 - e.g. Catastrophic -> Extreme
- Strengths
 - AVRMA risk level is assigned.
 - No flawed likelihood assessment.
 - Consistency.
- Weaknesses
 - Not all shortfalls are equal.
 - Potentially inaccurate.
 - Likelihood may be reverse engineered anyway.





Option 3: Uncertainty

- Overview
 - Adapt AVRMM to allow degrees of uncertainty to be assigned to software derived risks.
 - Uncertainty: Minimal, Partial, Extensive, Pervasive, Total.
 - Map consequence and uncertainty to risk level.
- Strengths
 - Technically correct.
 - AVRMM risk level still assigned.
- Weaknesses
 - Change to AVRMM required.
 - Potential for uncertainty to be considered equivalent to likelihood.
 - Uncertainty may not be well understood by operators.





Option 4: Pass/Fail

- Overview
 - Do not assign a risk level, simply state whether software has met an acceptable benchmark or it hasn't.
 - Same approach as FAA.
- Strengths
 - No ambiguity, technically correct.
- Weaknesses
 - Not very helpful.





How can we measure software risk?





If we have to assign a likelihood...

- Some of the above options, including the current approach, require a likelihood to be assigned.
- How do we do this?
 - No sound engineering basis?
- DGTA has done this several times, following learnt:
 - the best method for estimating likelihood will vary depending on the data available (horses for courses)
 - the method is subjective (so there will be discussion amongst the informed) and has an element of black magic (so will not be well understood)
- This is not a path the ADF should go down, but recently we have had to.





An Example

- Competing tenders for provision of primary flight display software.
- Required to meet DO-178B Level A.
- Option 1:
 - Meets all Level A objectives except source to object traceability.
- Option 2:
 - Only meets the Level D objectives (i.e. functional performance)
- No known faults with either.
- Which one presents less safety risk?





An Example

- If you said Option 1, why?
 - Consequence is the same (Catastrophic).
 - Have you made a distinction based on integrity?
 - Is integrity a substitute for likelihood?
- The fact that Option 1 and Option 2 can be distinguished suggests that there is a second dimension to software risk.





How can we do it?

- Depends on data
- Option A: Use Assurance Levels
- Option B: Scale of Shortfalls
- Option C: Best Guess of Experts
- Option D: Pessimistic Expert Opinion





Option A: Software Levels

- Overview
 - DO-178B provides a scale for measuring confidence in software behaviours (i.e. inverse of uncertainty).
 - Map DO-178B levels to likelihoods/uncertainties.
 - e.g. Level A -> Extremely Improbable
- Strengths
 - Robust, objective.
 - Not technically correct, but probably not grossly incorrect.
- Weaknesses
 - Requires substantial data.
 - Unlikely to provide sufficient resolution:
 - Lots of software is “Level D+”





Option B: Scale of Shortfalls

■ Overview

- Rather than use DO-178B strictly, rank relative importance of objectives.
- Examples:
 - Likely: Most or all objectives have not been satisfied and there are system design flaws or known failure conditions.
 - Probable: Most or all objectives have not been satisfied.
 - Possible: A number of key objectives have not been satisfied.

■ Strengths

- Less data than Option A, better resolution, more considered.

■ Weaknesses

- Still requires substantial data, somewhat subjective.





Option C: Best Guess

- Overview
 - Rely on experts to make a best estimate (non-conservative)
- Strengths
 - May produce the best estimates
 - Minimal data requirements
 - Partial Credit
- Weaknesses
 - Subjective!
 - Tendency to underestimate
 - Not consistent with best practice
 - System should be considered unsafe until proven otherwise





Option D: Conservative Guess

- Overview
 - Also relies on experts, but the perspective is different
 - Conservative instead of best guess
- Strengths
 - As above, but adheres to best practice.
- Weaknesses
 - As above, but tendency to overestimate.





Where to from here?





Plan of Attack

- As this issue keeps coming up, the ADF needs a defined process that:
 - is (relatively) defensible,
 - ensures stakeholders are aware of shortfalls, and
 - meets stakeholder needs.
- DGTA is a stakeholder, but not the key one.
- Key driver is decision authorities: what do they need from the technical community?
- To that end, DGTA are drafting a paper aimed at eliciting stakeholder requirements.
- Once we have them, we will look at formalising a process.





What should you do from here?

- Don't use any of the techniques described in this presentation!
 - If you need these techniques, it means something in your project has gone horribly wrong.
 - Your first preference should be to avoid the need by satisfying the four airworthiness requirements for aviation software.
- If you have to:
 - Consult with SCI-DGTA early.





Conclusions

- The ADF may be required to trade software risk for cost, schedule and capability.
- Making a decision on this trade requires risks to be analysed.
- Current risk management processes require a likelihood to be assigned.
- Likelihoods cannot be correctly assigned when the source of risk is unknown behaviours in software.
 - But some software is better than others, so there is more to software risk than consequence.
- But, we may be able to make an estimate that is reasonable enough to inform a risk treatment decision.
 - Though it would not be good enough to argue for safety.
- Doing this is a last resort.





Questions?



??????????

Likelihood	Consequences				
	Catastrophic	Critical	Major	Moderate	Minor
Likely	Extreme	Very High	High	Medium	Medium
Probable	Extreme	Very High	High	Medium	Medium
Possible	High	High	Medium	Medium	Low
Improbable	Medium	Medium	Medium	Low	Low
Rare	Low	Low	Low	Low	Low

